

# Running OpenFOAM on Android - A Workflow Tutorial with a Simple Case Study

Komal Kedarnath G  
Mechanical Engineer, Researcher  
Kolar Gold Fields, Karnataka, India

## Abstract

Android is at its core “just another Linux distribution”. Anything you can do on a typical Linux desktop can be achieved on android too, to a good extent. The technology is evolving as well, to diminish the gap between handheld smartphones and desktop personal computers. It is therefore common to find a large population in India owning Android phones with at least 4 cores but don’t own computers that are equally or more powerful. This tutorial is intended to set a framework that can be adopted by many and built upon further. This tutorial will consist of two parts – 1. Setting up Android devices with as few extra applications as possible to run OpenFOAM. 2. Running a simple elbow case in it. A few extended applications will be hinted upon for moderately advanced users. Anyone with knowledge of basic Linux can pick up on them to further their exploration and learn through it.

## 1 Introduction

OpenFOAM is a versatile tool with a great community. There is too much power in the hands of the user due to the open source code, leaving so much room for experimentation. The application can prove to be as light or as heavy on the computational resources as required by the case being run. One such possibility is explored here by running a successful case of OpenFOAM on an Android smartphone. Android being essentially a heavily modified Linux operating system, there is just a little effort required to set up such an environment. This is expected to lead to more experimentation on the viability and probable applications of running OpenFOAM in such ARM devices.

## 2 Problem Statement

To set up and run OpenFOAM on any android phone. For this session we use Samsung SM-M315F/DS with Android 11/One UI 3.1. It has an octacore processor and 6 Gb RAM. We will be focusing on the elbow case from the in-compressible section in the tutorials folder.

### 3 Governing Equations and Models

$$\text{ContinuityEquation} : \nabla \cdot \mathbf{u} = 0 \quad (1)$$

$$\text{MomentumEquations} : \frac{\partial u}{\partial t} + \nabla \cdot (uu) = -\frac{1}{\rho} \nabla \bar{p} + \nabla \cdot v [2S] \quad (2)$$

$$\text{Pressure} : p = \frac{1}{\rho} \bar{p} \quad (3)$$

## 4 Simulation Procedure

### 4.1 Setting up terminal environment

Android is essentially Linux, albeit a heavy modified version of Linux. Hence all we need is a terminal emulator application and we can practically replicate the desktop experience on an android phone as the effectiveness and universal nature of terminals is well known to all OpenFOAM users.

For this, one can use the application called Termux, a powerful and open-source terminal emulator with a great community. It hosts many scientific packages, hence the builds are available quickly upon release.

Termux can be found on playstore here - **Termux on Playstore**

This version of Termux on playstore is not up-to date. But it surely works for the results we wish to achieve. The most updated version is available at Fdroid here - **Termux on Fdroid**. It can be either installed via the Fdroid App or the APK can be downloaded from the site and installed on the android phone manually

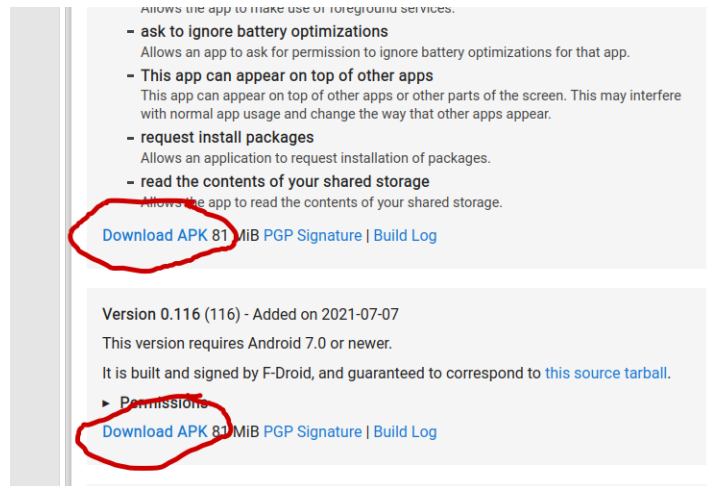


Figure 1: Download option on F-Droid website

Once Termux is installed, we have just opened up unlimited possibilities on the smartphone. Termux terminal allows us to use android just like one would use a normal Linux terminal.

A couple of things require attention right after installation. That is getting some basic programs and setting up Termux to access phone storage. This helps us load case files or examples onto the storage and access them in Termux to run OpenFOAM operations on them.

To set up storage access, type the following commands in Termux app. [1]

```
$ pkg update  
$ pkg upgrade  
$ termux-setup-storage
```

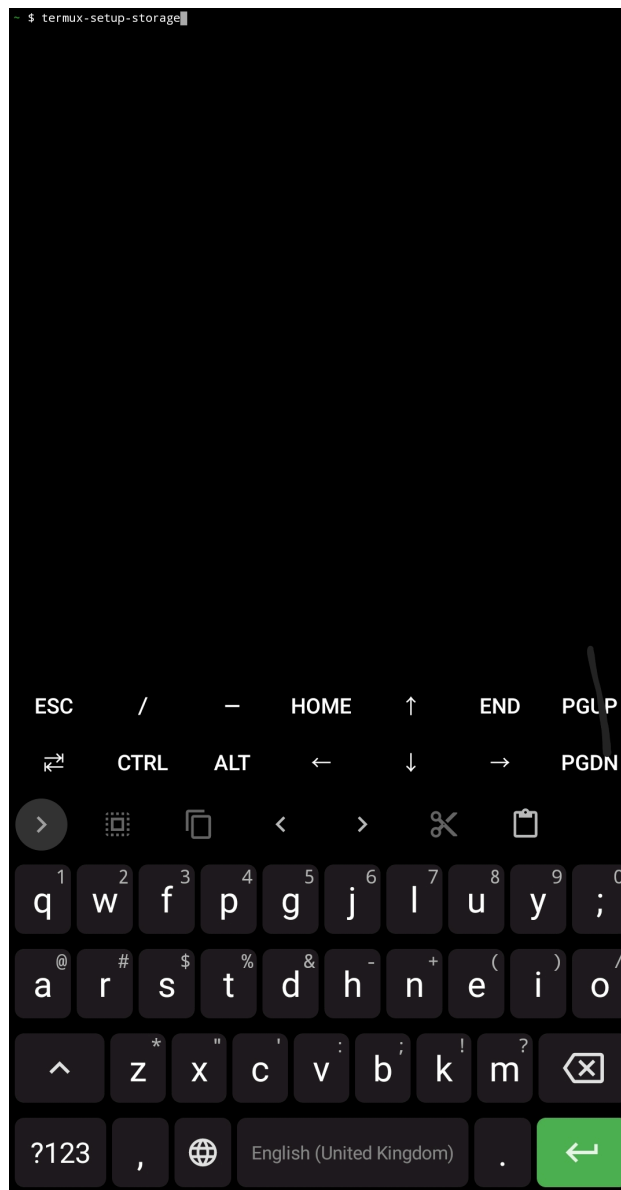


Figure 2: Running the 'termux-setup-storage' command in termux terminal

Once termux-setup-storage is run, a prompt will pop up asking for storage permission. Allow it to access storage. Now typing "ls" will show that there is "storage" folder that is accessible. The Documents, Downloads, Music, Pictures etc folders in the default location can all be accessed via the storage folder.

## 4.2 Installation of OpenFOAM and other Pre-Requisites

To install OpenFOAM type the following command into Termux.

```
$ pkg install openfoam
```

Currently the Termux package manager installs OpenFOAM V2106 which is equivalent to V9. There is no way to choose an older version apart from changing the main repositories, which is not covered in this study. Once the process is complete, restart the app. Now source the OpenFOAM files using the following command. Remember you will have to do it every time you start the app. You can always add the command to an auto-start script so it is run automatically.

```
$ source $PREFIX/opt/OpenFOAM-v2106/etc/bashrc
```

All the tutorials can be found here -

```
$ $PREFIX/opt/OpenFOAM-v2106
```

To check if the configuration of OpenFOAM is done correctly run icoFoam, If you get the icoFoam message on terminal then you have successfully set up OpenFOAM. Along with OpenFOAM some other programs are needed.

Let us install gnuplot for graphing purposes. Run the following commands to install gnuplot and vim.

```
$ pkg install gnuplot
$ pkg install vim
```

The following programs are needed for graphical experience. How to use graphical x11 applications will be covered in the 'running case' section. However the programs can be installed by following the commands below.

```
$ pkg install x11-repo
$ pkg install tigervnc
$ vncserver -localhost
```

[2] At first time, you will be prompted for setting up passwords. Note that passwords are not visible when you are typing them and maximum password length is 8 characters.

We need to instal VNC app (virtual network computing). You can use any VNC app, for this session multivnc is preferred which can be found here - Playstore, Fdroid.

Now to make things easier in the graphical environment, a Desktop environment (DE) even can be installed to be used on top of Termux. Many DEs even xfce is available but for this session OpenBox is preferred.

The following commands will help install OpenBox

```
$ pkg install openbox pypanel xorg-xsetroot  
$ vim ~/.vnc/xstartup
```

Change the contents of the file as given below

```
#!/data/data/com.termux/files/usr/bin/sh  
  
# Start Openbox.  
openbox-session &
```

Next, do the following as well

```
$ vim ${PREFIX}/etc/xdg/openbox/autostart
```

Change the contents as follows

```
# Make background gray.  
xsetroot -solid gray  
  
# Launch PyPanel.  
pypanel &
```

All that is needed is installed and configured. The running of cases in OpenFOAM can now be explored.

### 4.3 Running Cases

To run cases the examples can be copied from the default path as mentioned above or can be taken from another PC or Internet etc.. Our goal is to have the files with us on the storage so we can run operations on it. Here the elbow case from in-compressible examples is copied from examples folder in PC OpenFOAM directory. It is copied to the downloads folder in default home location where there are Documents, Downloads, etc.. folders.

For this study the default parameters are left as-is but they can be changed to meet the simulation needs. Run the following commands in the elbow folder.

```
$ fluentMeshToFoam elbow.msh  
$ icoFoam> log
```

```
or even run in parallel if you have cores to spare as follows.  
$ mpirun -np <no.of cores> <solver> -parallel > log.filename &
```

Notice that we are writing the output of icoFoam to a log file. This is very essential as most of the post-processing will be carried through the log file. The errors can always be read in the log file. If the output is preferred to be printed to the terminal, one can always run "\$ cat" command on the log file after the above command.

This marks the end of running OpenFOAM. Any case can be edited and modified using vim and run similarly.

## 4.4 Post-Processing

Post-processing will be covered in two sections. Non-graphical and Graphical methods. Running of OpenFOAM on android is covered so far. But unfortunately there is no Paraview port for arm devices, hence it cannot be installed on Termux. It can still be run on Android through Dex (special purpose software available on high end samsung devices). It is not covered here as it would oppose the intentions of running on any android device. Well if we don't have paraview, how to carry the post-processing?. This is one limitation of OpenFOAM Android. The only post-processing available is through the internal functions of OpenFOAM. An instance will be covered in the next sections.

## 4.5 Post-Processing (non graphical):

The log file is a very useful piece of text. There are multiple built in function in OpenFOAM that can be used in post-processing but only one method will be covered here. This method is first covered entirely in 'Terminal' and then a 'graphical method' is shown as well. First run foamLog function on the log file.

```
$ foamLog log
$ cd logs
```

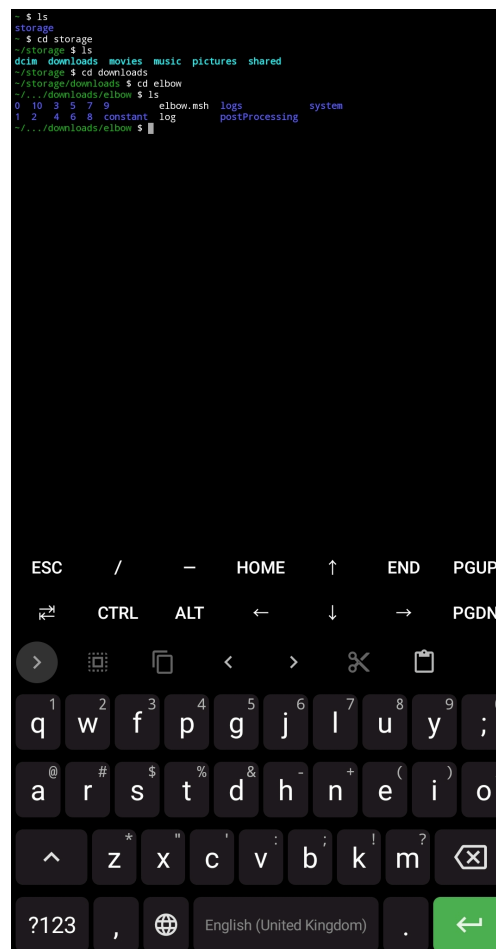


Figure 3: Output of 'ls' command in the root folder after running the 'foamLog' command

```

~ $ ls
storage
~ $ cd storage
~/storage $ ls
dcim  downloads  movies  music  pictures  shared
~/storage $ cd downloads
~/storage/downloads $ cd elbow
~/.../downloads/elbow $ ls
0  10  3  5  7  9      elbow.msh  logs      system
1  2  4  6  8      constant  log      postProcessing
~/.../downloads/elbow $ cd logs
~/.../elbow/logs $ ls
CourantMax_0  UyFinalRes_0      contGlobal_1      pFinalRes_2      pIters_3      p_4
CourantMean_0  UyIters_0          contLocal_0       pFinalRes_3      pIters_4      p_5
Separator_0    Uy_0               contLocal_1       pFinalRes_4      pIters_5
Time_0         clockTime_0        executionTime_0    pFinalRes_5      p_0
UxFinalRes_0   contCumulative_0   foamLog.awk        pIters_0         p_1
UxIters_0      contCumulative_1   pFinalRes_0        pIters_1         p_2
Ux_0           contGlobal_0       pFinalRes_1        pIters_2         p_3
~/.../elbow/logs $

```

Figure 4: Output of 'ls' command in the 'logs' folder

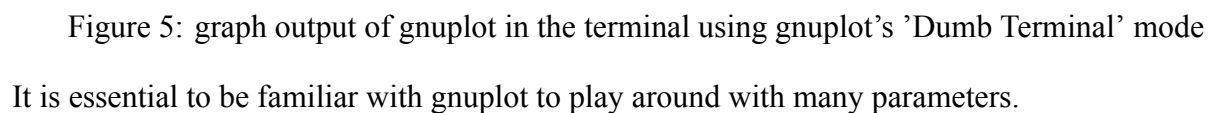
This will create a folder called logs where you have files for a number of variables, with data that can be plotted with gnuplot.

If you know the basics of gnuplot you can play around with the plotting and the variable files. An example is given below. Please note to set terminal to "dumb" this is important to draw graphs in terminal.

```

$ gnuplot
$ set terminal dumb
$ plot "CourantMax_0" using 1:2 with lines

```



## Port : 5901



Password : the password you set while configuring the vnc in earlier step  
Press connect. You can save the profile by clicking on save bookmark  
so you don't have to enter the details every time.

Once VNC server is connected, you will see a grey screen with a black panel at the bottom. Three mouse buttons at the bottom of the phone screen. You can touch and move your finger across the screen anywhere to move the mouse cursor and use the buttons at the bottom to emulate left click, right click and middle click of the mouse.

Now right click anywhere on the grey area. You will see a menu, click on terminal. This will open up a terminal in x11 window.

On the top right corner you will see a hamburger menu, click on that and select toggle keyboard to get a keyboard. You can now do a lot of operations on the Terminal just like you would on a PC.

Typing "ls" command will show storage, navigate to storage and then to the case folder and to logs folder. Type gnuplot here. This will start gnuplot in the current folder and you can plot whatever variables you want. Example below.

```
$ gnuplot  
$ plot "CourantMax_0" using 1:2 with lines
```

Illustrations Follow.

```
~/.../elbow/logs $ vncserver -localhost
New 'localhost:2 ()' desktop is localhost:2
Starting applications specified in /data/data/com.termux/files/home/.vnc/xstartup
Log file is /data/data/com.termux/files/home/.vnc/localhost:2.log
~/.../elbow/logs $
```

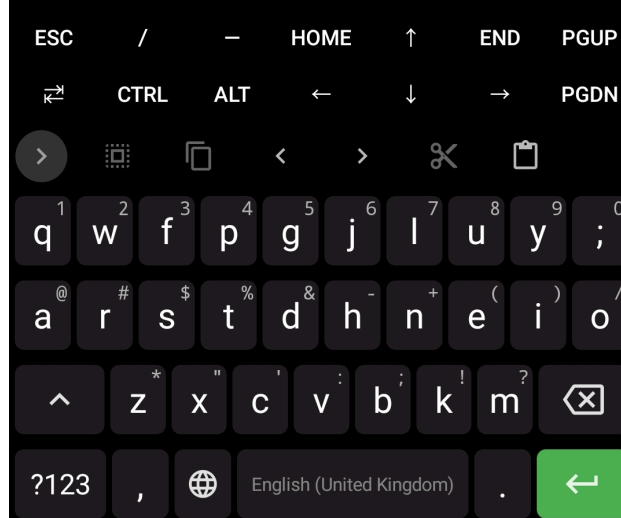





Figure 6: Output of the 'vncserver -localhost' command

**MultiVNC**   

Discovered Servers

Bookmarks

New Connection

Address

127.0.0.1

Port

5901

Color Mode

24-bit color (4 bpp) ▼

Username

For Windows/Mac authentica

Password

..... ☐ Keep

ID On Repeater

Server ID when connecting to

SAVE BOOKMARK

Connect

Figure 7: UI of 'multivnc' front page with "address:127.0.0.1 and port:5901"

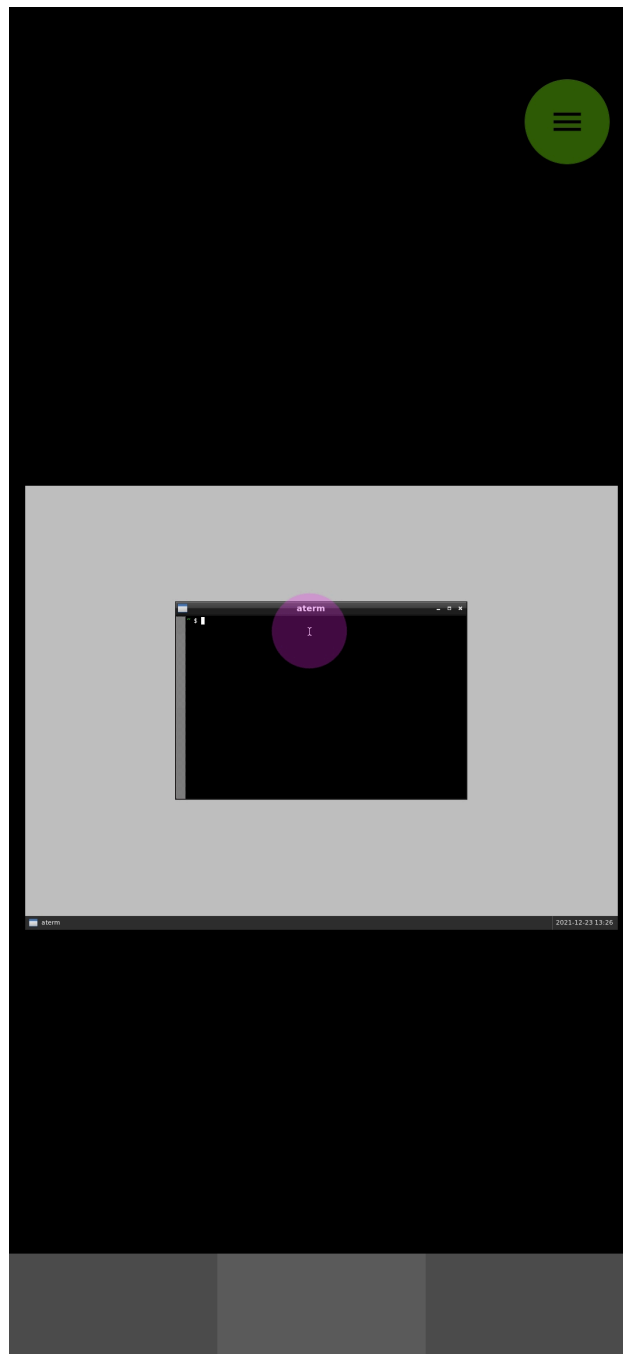


Figure 8: UI of the 'multivnc' app after connecting to the localhost address given above and with a terminal opened using the right-click menu. Three mouse buttons on the bottom and hamburger menu on top-right

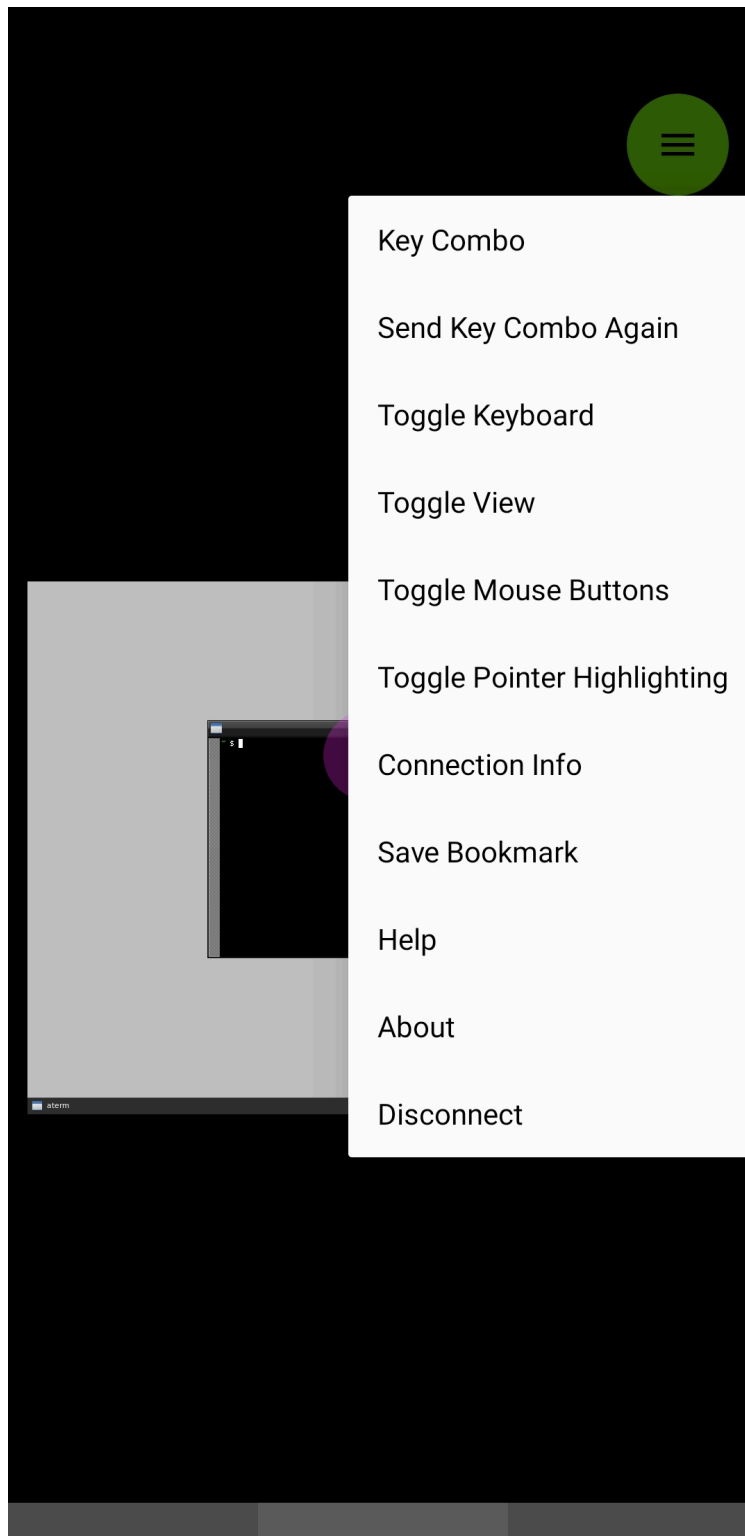


Figure 9: Contents of the hamburger menu in the 'multivnc' app

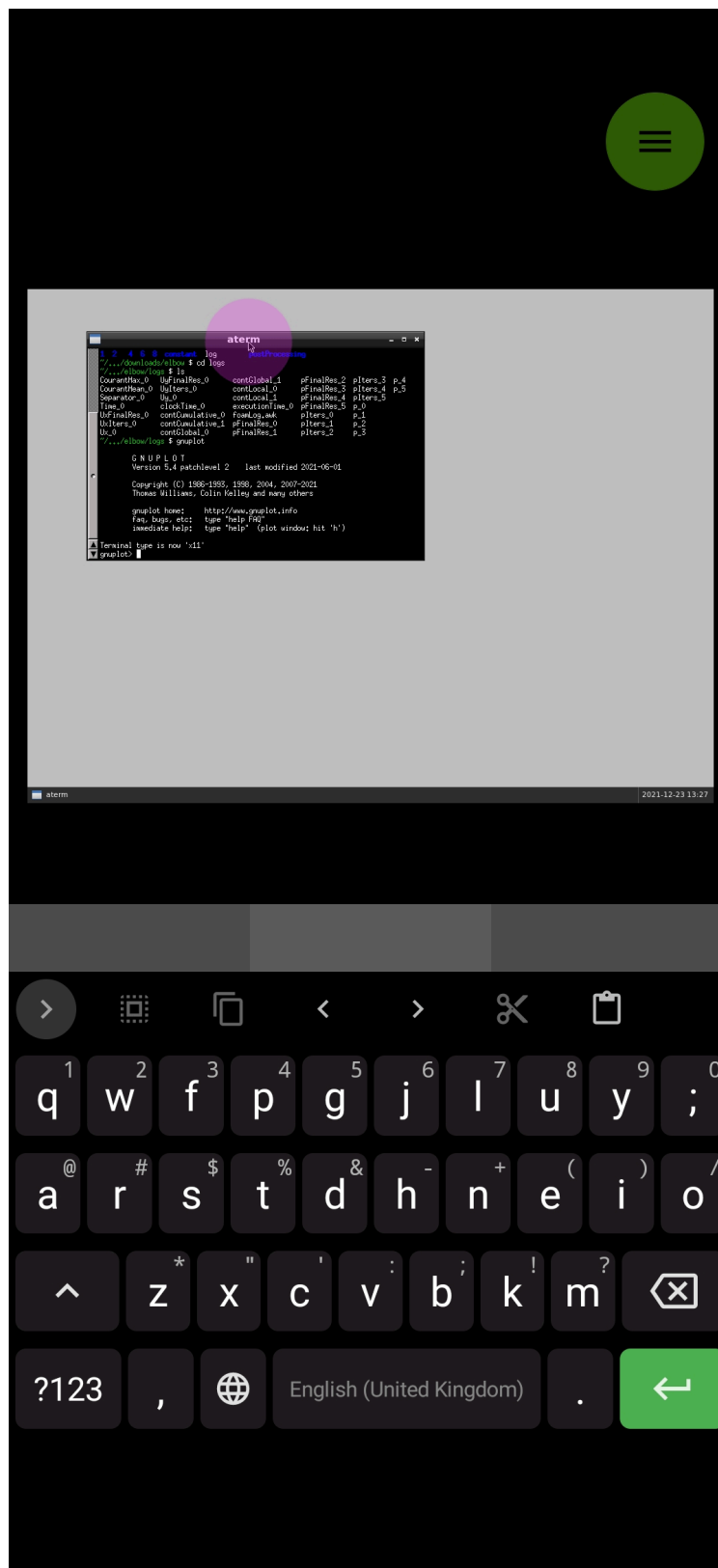


Figure 10: Running gnuplot in terminal in the 'multivnc' app

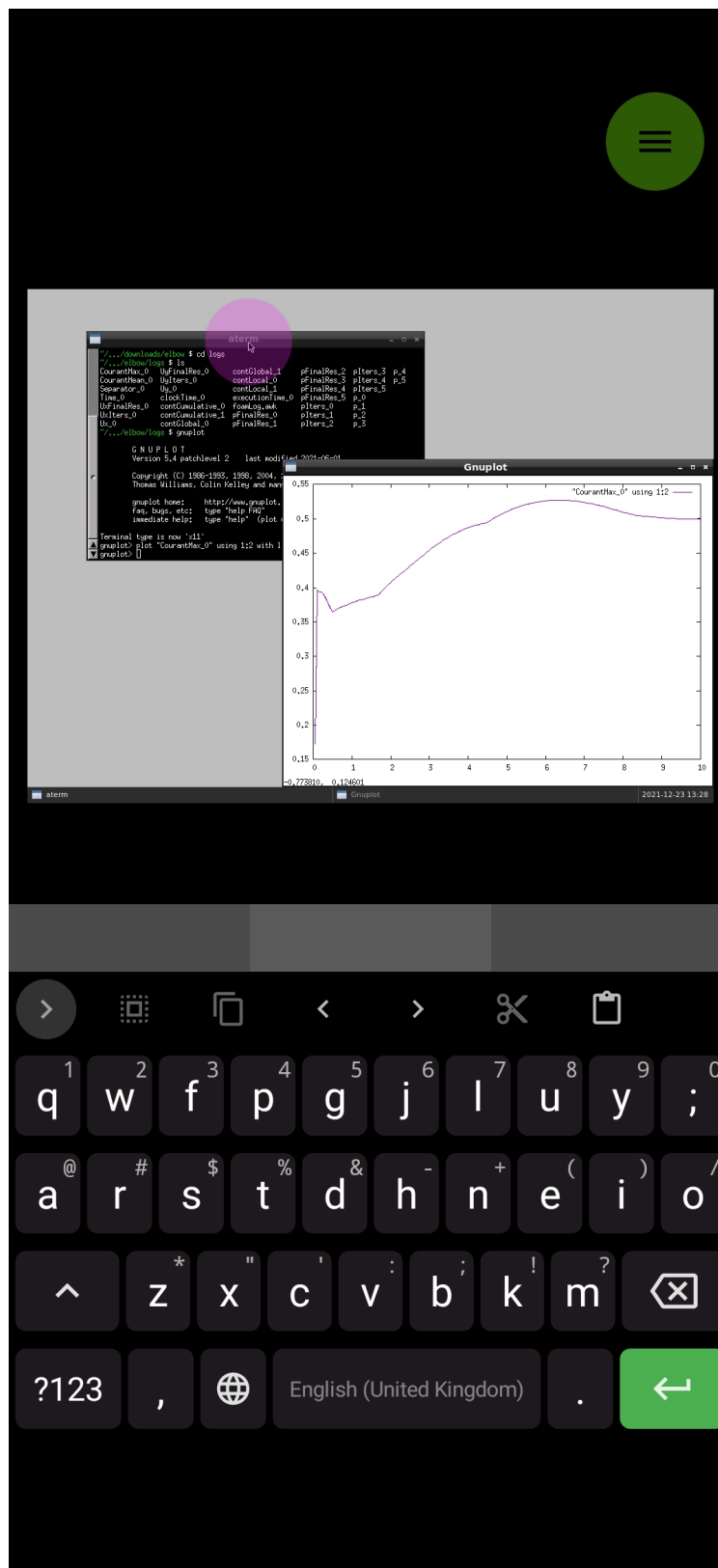


Figure 11: Graph output of gnuplot. This is the same graph obtained in non-graphical section within the terminal, this time the output is completely graphical.

A new window will appear with the graph plotted on it. This is different to the terminal graph method as that would draw the graph in the terminal itself, whereas in this method we get to see the whole graph in a new window with all colors as we do normally in a PC.

## 5 Results and Discussions

This concludes the workflow tutorial. There are many areas that can be further explored and developed upon. Our smartphones are getting smarter and powerful each day. Especially, Android due to its Linux roots has unlimited possibilities. It is only through creativity can we liberate our minds. This method was discovered through a phase of hardship, I compel everyone reading this report to embrace the limitations to build something beautiful.

## References

- [1] <https://wiki.termux.com/wiki/Termux-setup-storage>
- [2] [https://wiki.termux.com/wiki/Graphical\\_Environment](https://wiki.termux.com/wiki/Graphical_Environment)